Elementos de Lógica Digital II

Aula 2 – Introduction to VHDL

Prof. Vanderlei Bonato - vbonato@icmc.usp.br

Summary

- History
- VHDL Structure
- Sequencial and Parallel Execution
- Signal and Variable

Concepts

- **VHDL is the VHSIC (Very High Speed Integrated Circuit) Hardware Description Language**
- **VHDL is an international standard specification language for describing digital hardware used by industry worldwide**
- **VHDL enables hardware modeling from the gate to system level**
- **VHDL provides a mechanism for digital design and reusable design documentation**

History of VHDL

- **Launched in 1980**
- **Aggressive effort to advance state of the art**
- **Object was to achieve significant gains in VLSI technology**
- **Need for common descriptive language**
- **In July 1983, a team of Intermetrics, IBM and Texas Instruments were awarded a contract to develop VHDL**

History of VHDL

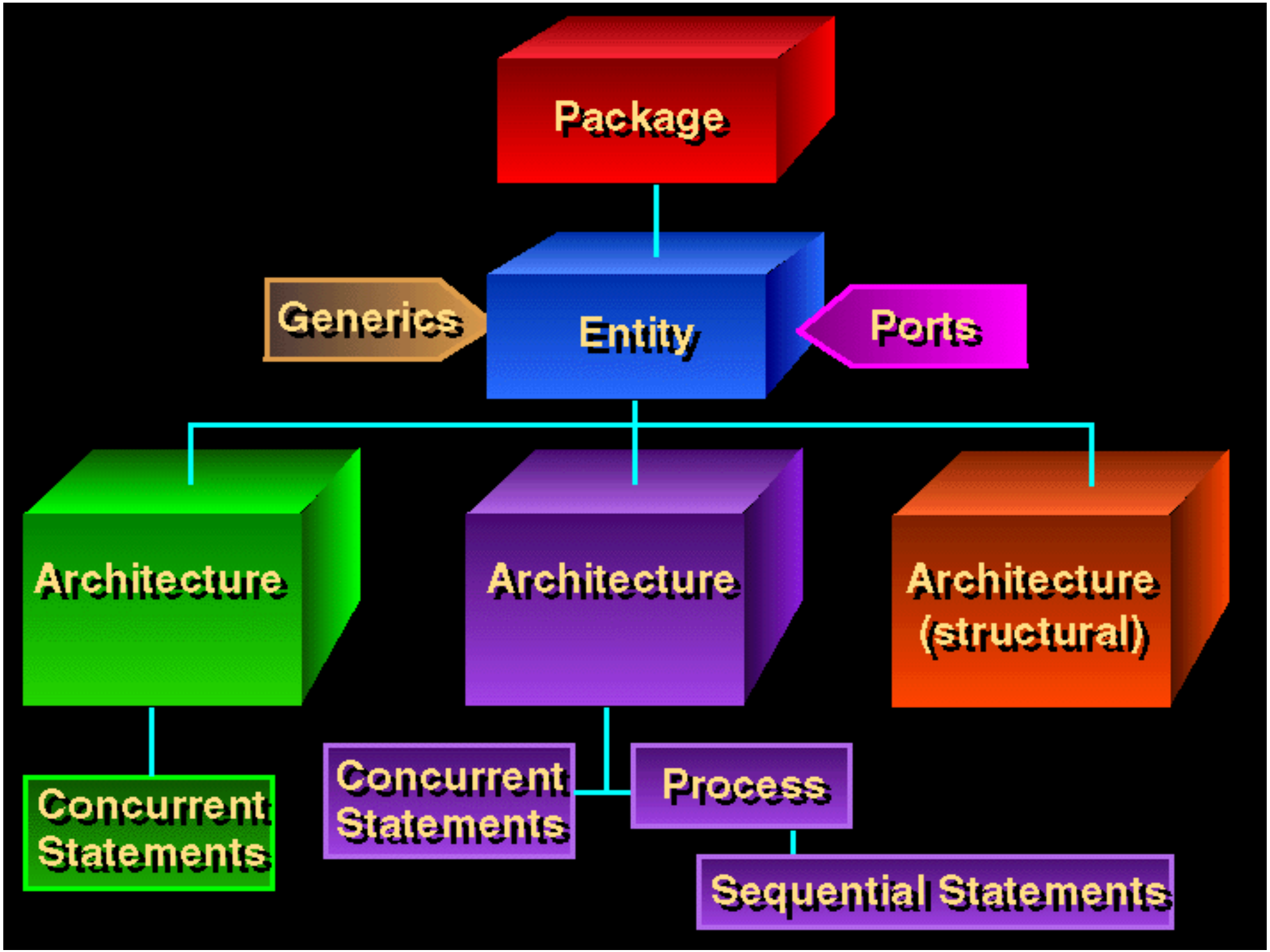
- **In August 1985, the final version of the language under government contract was released: VHDL Version 7.2**
- **In December 1987, VHDL became IEEE Standard 1076-1987 and in 1988 an ANSI approved standard**
- **In September 1993, VHDL was restandardized to clarify and enhance the language (IEEE Standard 1076-1993)**
- **Since then there has been many other VHDL standard revision**

How about Quartus II

- **The Quartus II software supports a subset of the constructs defined by the IEEE Std 1076-1987, and IEEE Std 1076-1993, and IEEE Std 1076-2008**
  - **It supports only those constructs that are relevant to logic synthesis**
- **The Quartus II 11.1 software contains support for VHDL 2008: IEEE Std 1076-2008**
- **The Quartus II software also supports the packages defined by these patterns**

Why Use VHDL?

- **Provides technology independence**
- **Describes a wide variety of digital hardware**
- **Eases communication through standard language**
- **Allows for better design management**
- **Provides a flexible design language**


Package
Generics
Entity
Ports
Architecture
Architecture
Architecture (structural)
Concurrent Statements
Concurrent Statements
Process
Sequential Statements

VHDL Design Process

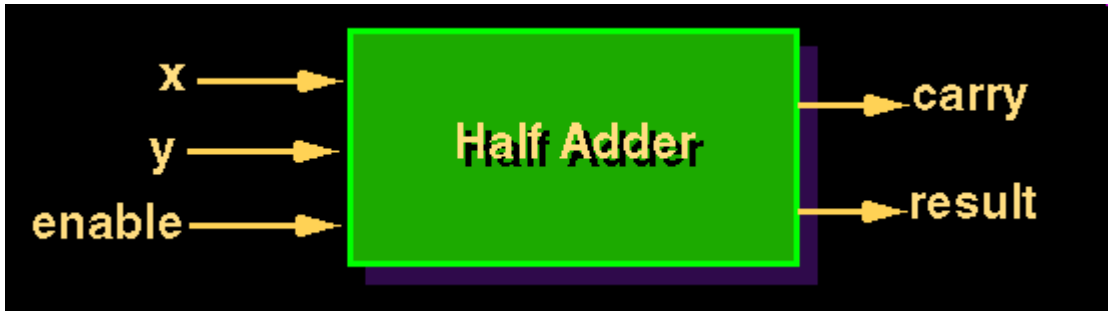
- **Problem: design a single bit half adder with carry and enable**
- **Specifications**
  - **Passes results only on enable high**
  - **Passes zero on enable low**
  - **Result gets x plus y**
  - **Carry gets any carry of x plus y**

Entity Declaration

- **An entity declaration describes the interface of the component**
- **PORT clause indicates input and output ports**
- **An entity can be thought of as a symbol for a component**
- **Generics may be added for readability, maintenance and configuration**

Entity Declaration

```
ENTITY half_adder IS

      PORT (x, y, enable: IN bit;

                carry, result: OUT bit);



END half_adder;
```



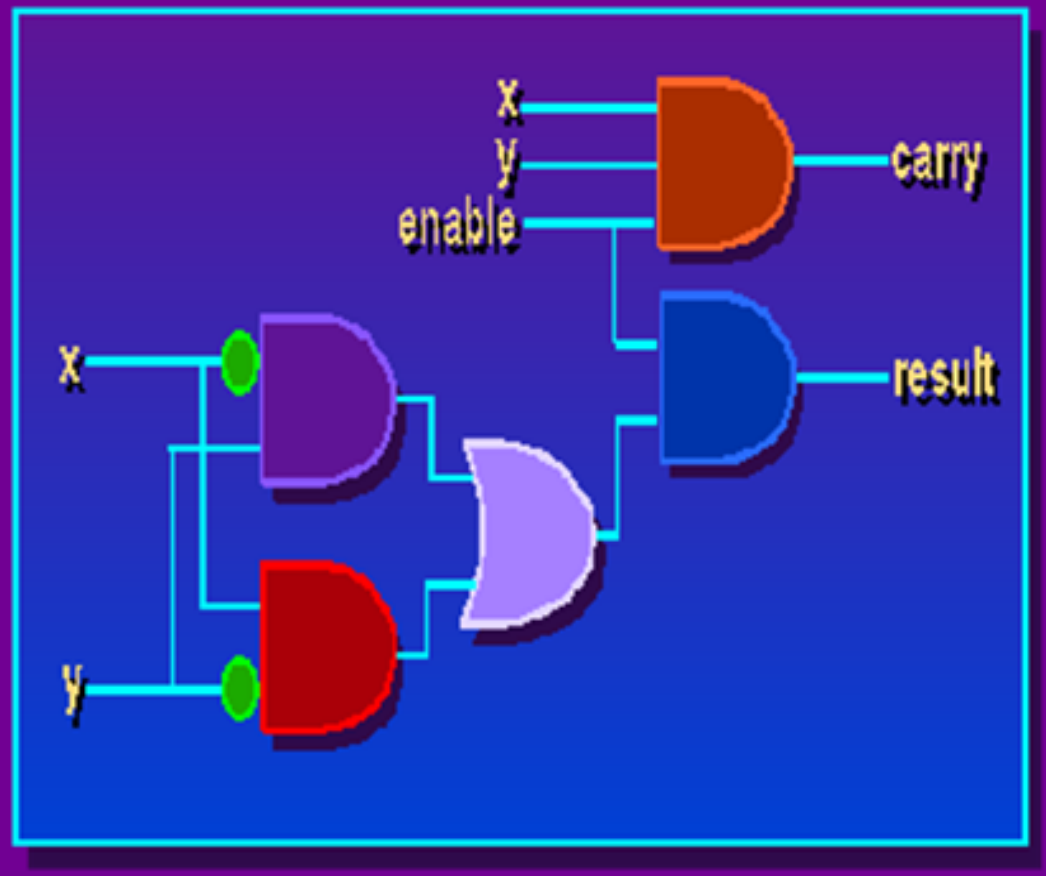
Architecture Declaration

- **Architecture declarations describe the operation of the component**

- **Many architectures may exist for one entity, but only one may be active at a time**

- **An architecture is similar to a schematic of the component**

```
ARCHITECTURE behavior1 OF

half_adder IS BEGIN

    PROCESS (enable, x, y)

    BEGIN

    IF (enable = '1') THEN

      result <= x XOR y;

      carry <= x AND y;

    ELSE

      carry <= '0'

      result <= '0';

    END PROCESS;

END behavior1;
```



Packages and Libraries

- **User defined constructs declared inside architectures and entities are not visible to other entities**
  - **Subprograms, user defined data types, and constants can not be shared**
- **Packages and libraries provide the ability to reuse constructs in multiple entities and architectures**
- **An important VHDL library is the IEEE library. This package provides a set of user-defined datatypes and conversion functions that should be used in VHDL designs.**
  - **LIBRARY ieee;**
  - **USE ieee.std_logic_1164.ALL;**

Sequential and Concurrent Statements

- **VHDL provides two different types of execution: sequential and concurrent**
- **Different types of execution are useful for modeling of real hardware**
  - **Supports various levels of abstraction**
- **Sequential statements view hardware from a "programmer" approach**
- **Concurrent statements are order-independent and asynchronous**

Sequential Statements

- **Sequential statements run in top to bottom order**
- **Sequential execution most often found in behavioral descriptions**
- **Statements inside PROCESS execute sequentially**

Concurrent Statements

- **All concurrent statements occur simultaneously**
- **How are concurrent statements processed?**
- **Simulator time does not advance until all concurrent statements are processed**
- **Some concurrent statements**
  - **Block, process, assert, signal assignment, procedure call, component instantiation**

VHDL Processes

- Assignments executed sequentially
- Sequential statements
  - {Signal, variable} assignments
  - Flow control
    - if <condition> then <statements> else <statements> end if;
    - for <range> loop <statements> end loop;
      while <condition> loop <statements> end loop;
    - case <condition> is when <value> => <statements>;
      when <value> => <statements>;
      when others => <statements>;
      end case;
  - Wait on <signal> until <expression> for <time>;
  - Assert <condition> report <string> severity <level>;

VHDL Processes

- A VHDL process statement is used for all behavioral descriptions

```
[process_label :] PROCESS

[(sensitivity_list)]


  process_declarations


BEGIN


  process_statements


END PROCESS [process_label];
```

Process Example - Carry Bit

```
Carry: PROCESS(A, B, Cin)
 BEGIN
  IF (A = '1' and B = '1') THEN
       Cout <= '1';
  ELSIF (A = '1' and Cin = '1') THEN
       Cout <= '1';
  ELSIF (B = '1' and Cin = '1') THEN
       Cout <= '1';
  ELSE
       Cout <= '0';
  END IF;
END PROCESS Carry;
```

A Design Example: 8-bits Register with asynchronous clear

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY reg8 IS
    PORT (clock, rst : IN BIT;
          D: IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
          Q: OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END reg8;

ARCHITECTURE behavior OF reg8 IS

BEGIN
    register8: PROCESS (clock,rst)
    BEGIN
        IF rst = '0' THEN
            Q <= "00000000";
        ELSIF clock'EVENT AND clock ='1' THEN
           Q <= D;
        END IF;
    END PROCESS register8;
END behavior;
```

A Design Example: 2-bits Counter

```
ENTITY count2 IS
    PORT (clock : IN BIT;
          q1, q0: OUT BIT);
END count2;

ARCHITECTURE behavior OF count2 IS

BEGIN
    count_up: PROCESS (clock)
        VARIABLE count_value: NATURAL := 0;

    BEGIN
        IF clock='1' THEN
            count_value := (count_value+1) MOD 4;
            q0 <= bit'val(count_value MOD 2);
            q1 <= bit'val(count_value/2);
        END IF;
    END PROCESS count_up;
END behavior;
```

Signals vs Variables

- **Variables**
  - **Used for local storage of data**
  - **Generally not available to multiple components and processes**
  - **All variable assignments take place immediately**
  - **Variables are more convenient than signals for the storage of data**
  - **Variables may be made global**
- **Signals**
  - **Used for communication between components**
  - **Signals can be seen as real, physical signals**
  - **Some delay must be incurred in a signal assignment**

Assignments

```
ARCHITECTURE test1 OF

test_mux IS

    SIGNAL a : BIT := '1';

    SIGNAL b : BIT := '0';

BEGIN


...more statements...

   a <= b;

   b <= a;

...more statements...


END test1;
```

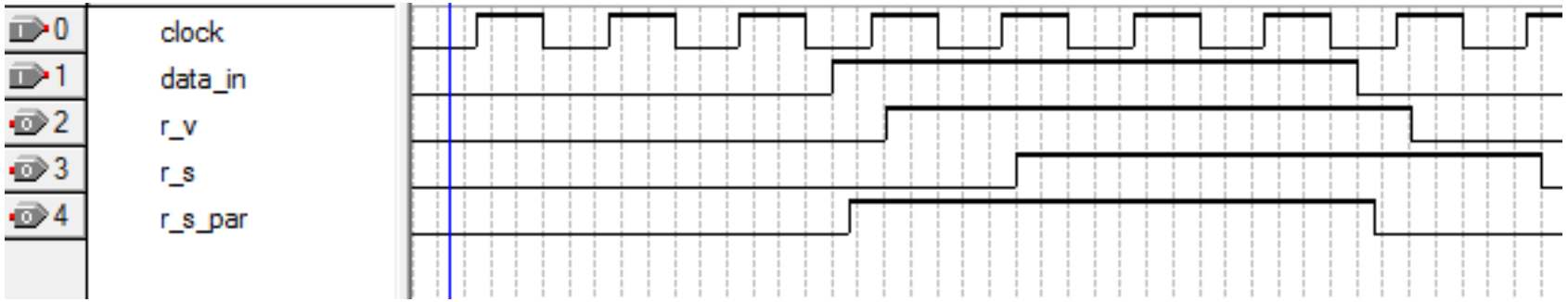
```
ARCHITECTURE test2 OF test_mux IS BEGIN

PROCESS (result)

   VARIABLE a : BIT := '1';

   VARIABLE b : BIT := '0';

BEGIN

...more statements...

   a := b;

   b := a;

...more statements...

   END PROCESS;

END test2;
```

Signal x Variable Behaviour

```
ENTITY aulavhdl IS
    PORT (clock, data_in : IN BIT;
          r_v, r_s, r_s_par: OUT BIT);
END aulavhdl;

ARCHITECTURE behavior OF aulavhdl IS
    signal a_s, a_s_par: BIT := '0';
BEGIN
    PROCESS (clock)
        variable a_v: BIT := '0';
    BEGIN
        IF clock='1' THEN
            a_v := data_in;
            r_v <= a_v;

            a_s <= data_in;
            r_s <= a_s;
        END IF;
    END PROCESS;
    a_s_par <= data_in;
    r_s_par <= a_s_par;
END behavior;
```

Signal x Variable Behaviour

- Percebam a diferença de comportamento do “signal” dentro e fora do processo!
- Quanto a “variable” não há surpresa, pois é utilizada somente dentro do processo



References

- Pedroni, Volnei A. Circuit Design with VHDL, MIT Press, 2004

- DARPA/Tri-Services RASSP Program
  - http://www.vhdl.org/rassp/

- Brown, S. and Vranesic, Z.. Fundamentals of Digital Logic with VHDL Design, 2nd Ed., P.939, 2005.